RSF Elektronik

USER MANUAL
IFC 430R

PC-PLUG-IN CARD

02/2013

CONTENTS

1. General Information 03
1.1 Important Information 03
1.2 Application 03
1.3 Items supplied 03

2. Specifications 04
2.1 Mechanical Dimensions and Ambient Conditions 04
2.2 PCI Bus 04
2.3 Counter Interface (X1) 04
2.4 I/O Interface (X2) 04
2.5 Counter Operating Modes 04
2.6 Latch Logic 04

3. Hardware 05
3.1 Component Location Diagram 05
3.2 Selecting the Encoder Operating Voltage 05
3.3 Pin Assignment X1 06
3.4 Pin Assignment X2 06
3.5 Input Wiring 07
3.6 Block Diagram 08

4. Address Allocation 09
4.1 Header Configuration 09
4.2 Local Address Allocation 09

5. Description of the Registers 10
5.1 System Control Register 10
5.2 Timer Register 10
5.3 Status Register 1 11
5.4 Status Register 2 11
5.5 Interrupt Register 12
5.6 Delay Timer for external Sync-In 13
5.7 Counter Load Register 14
5.8 Counter Latch Register 14
5.9 Counter Control Register 15
5.10 Counter Mode Register 15, 16

6. Instructions for Installation 17
6.1 Hardware Installation 17
6.2 Installation of the Drivers 17
6.3 Installation of the enclosed Test Software 17
6.4 Programming Examples 17

7. DLL Functions 18-26

1. GENERAL INFORMATION

The functions of a PCI bus are not described in this manual.

1.1 Important Information

- **Danger to components if these notes are not observed!**
- **Please observe the safety precautions according to DIN EN 100 015 when handling ESD components (electrostatic discharge)**
- **Only use antistatic packaging material.**
- **For mounting observe that the working place must be properly grounded.**
- **Do not engage or disengage any connectors while the unit is under power.**
- **Check the correct operating voltages of the encoders before inserting the jumpers of IFC 430R.**

1.2 Application

IFC 430R is a PC expansion board with PCI interface for the acquisition and evaluation of encoder signals.
It can also be used for all standard counting functions (event counter, frequency counter etc.).

1.3 Items Supplied

- IFC 430R PCI interface board
- Disc with demo program and driver software
- User Manual

2. SPECIFICATIONS

2.1 Mechanical dimensions and ambient conditions

- Dimensions (of the PCB) approx. 120 x 92 mm; width = 1 slot
- Maximum permissible ambient temperature +40°C
- 1 D-sub female terminal strip, 25-pin for the counter inputs
- 1 D-sub female terminal strip, 9-pin for the for I/O-signals

2.2 PC Bus

- PCI connector, 5 V, 32-bit, 2 x 60 pins
- Target interface (slave) as per specifications Rev. 2.1
- Bus clock frequency 40 MHz max.
- Current consumption at +5 V approx. 0.5 A, without encoders
- Power supply of the encoders: +5 V or +12 V from PCI power supply
  (current consumption depends on encoders connected)

2.3 Counter Interface (X1)

- 9 RS 422 or TTL inputs for 3 encoders with square-wave signals and reference mark
- Maximum input frequency: 5 MHz with delta signals (RS 422)
- 2 MHz with single-end signals
- 1 TTL input for interfering-signal monitoring
- Separate power supply lines for each encoder (0.5 A max. per encoder)

2.4 I/O Interface (X2)

- 6 inputs (3 to 30 V) that can be used as reference pulse inhibitors or as asynchronous latch signals
- 1 input (3 to 30 V) for synchronous latch of several channels
- 1 output (TTL) for cascading several cards

2.5 Counter Operating Modes

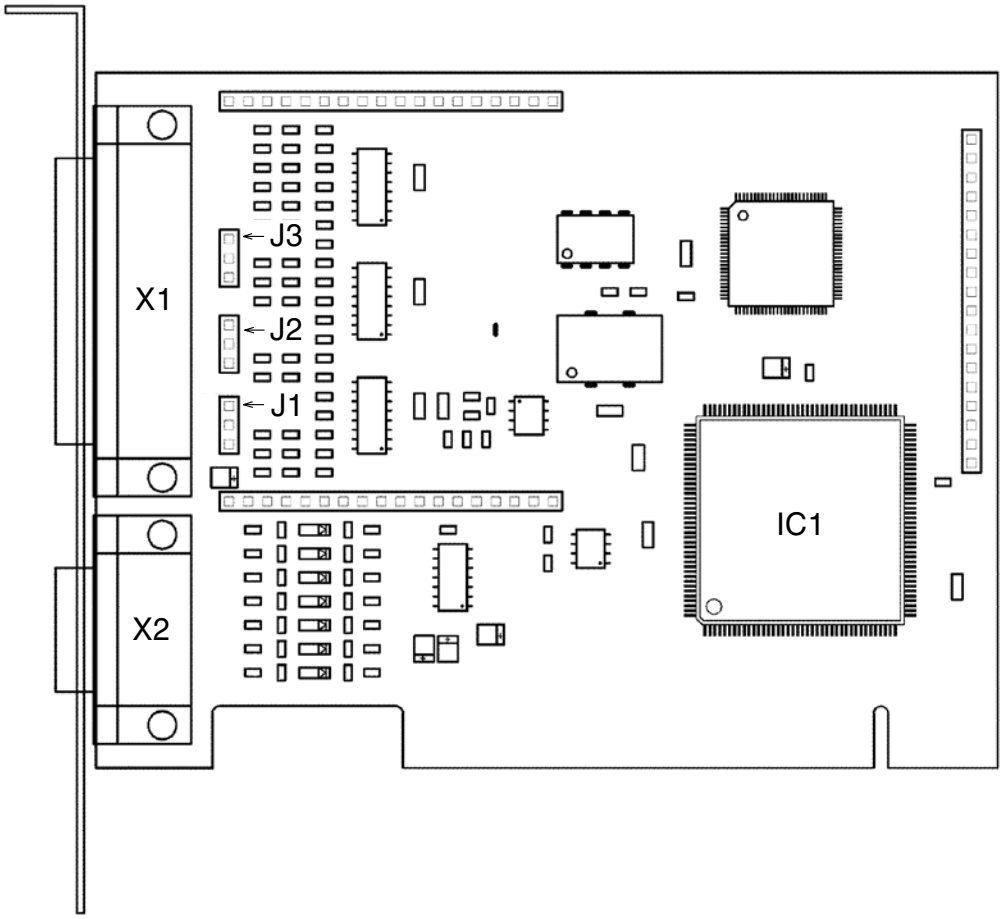
- Three counter channels, 32 bits each; one load and two latch registers for each channel.
- Counting of encoder square-wave signals with one-fold, two-fold or four-fold evaluation.
- Event counter with direction and clear input.
- Integral timer for measuring pulse width, frequency or speed.

2.6 Latch Logic

- Asynchronous latching of the counter values individually for each encoder channel via software, reference mark of the encoder or external hardware signal.
- Synchronous latch of several channels by software, timer or external signal.
- Output signal for cascading several cards; can be programmed for software, timer or external hardware synchronization.
- Latch time = two bus clock pulses = 60 ns at a clock frequency of 33.3 MHz

3. HARDWARE

3.1 Component Location Diagram



X1 = female D-sub terminal strip, 25-pin for counter interface
X2 = female D-sub terminal strip, 9-pin for switching and control signals
J1-J3 = jumper for the selection of the encoder operating voltage (5 V or 12 V)
IC1 = PCI interface

3.2 Selecting the Encoder Operating Voltage

The operating voltage of the encoders may be set individually for each channel to +5 V or +12 V by means of the jumpers J1 to J3. The selected voltage can be seen from the print on the board.

J1 = encoder channel 1
J2 = encoder channel 2
J3 = encoder channel 3

Note:
Observe that incorrect placement of the jumpers may destroy the encoder.
Default setting for the operating voltages is +5 V.

3.3 Pin Assignment X1

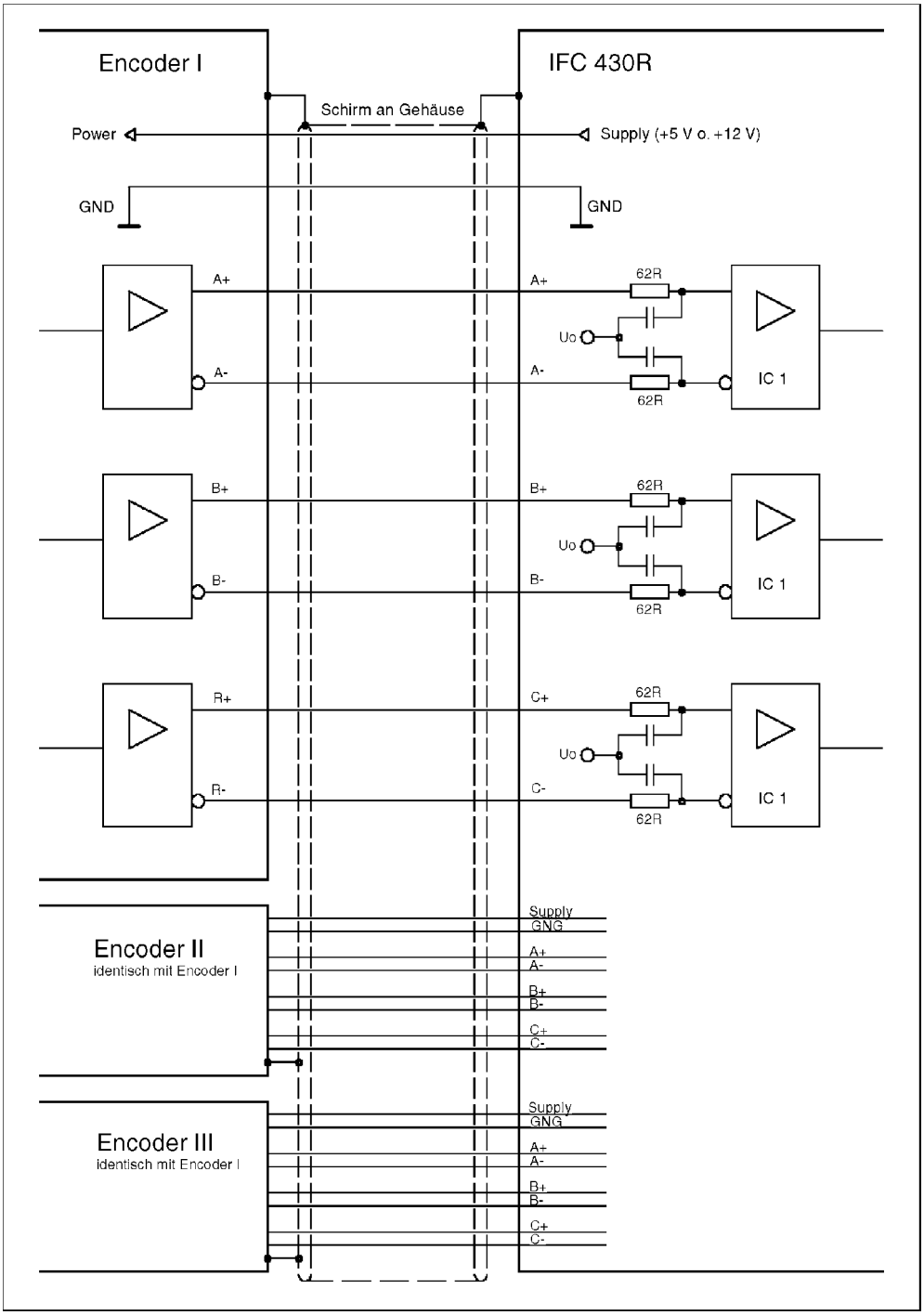
Pin	Signal
1	Channel 1 supply +5 V or +12 V (selected by J1)
14	Channel 1 GND
2	Channel 1 input A +
15	Channel 1 input A -
3	Channel 1 input B +
16	Channel 1 input B -
4	Channel 1 input C +
17	Channel 1 input C -
5	Channel 2 supply +5 V or +12 V (selected by J2)
18	Channel 2 GND
6	Channel 2 input A +
19	Channel 2 input A -
7	Channel 2 input B +
20	Channel 2 input B -
8	Channel 2 input C +
21	Channel 2 input c -
9	Channel 3 supply +5 V or +12 V (selected by J1)
22	Channel 3 GND
10	Channel 3 input A +
23	Channel 3 input A -
11	Channel 3 input B +
24	Channel 3 input B -
12	Channel 3 input C +
25	Channel 3 input C -
13	Interfering signal

3.4 Pin Assignment X2

Pin	Signal
1	GND
2	In 1
3	In 2
4	In 3
5	In 4
6	In 5
7	In 6
8	In Sync.
9	Out Casc.

3.5 Input Wiring

Encoder connection X1

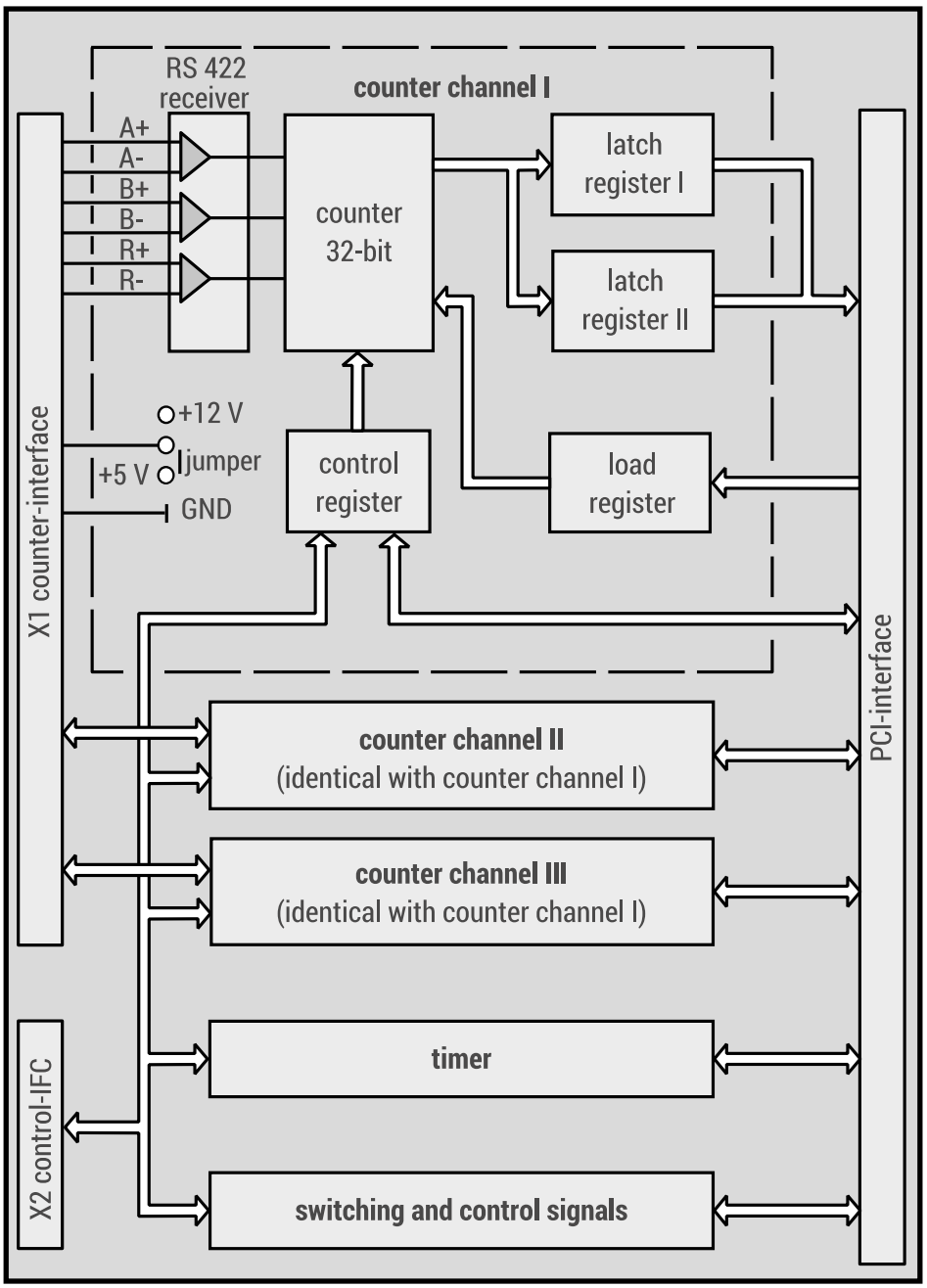


As line drivers all standard RS 422 drivers (such as MC3487, AM32LS31 etc.) may be used.

If there are no delta signals available, the inputs can also be wired as “single end inputs”; in this case one of the two delta inputs (normally –) remains open.

Encoders with 1 Vpp voltage interface may be operated like RS 422 encoders without any restrictions.

3.6 Block Diagram



4. ADDRESS ALLOCATION

PCI Interface

Interface: 32-bit PCI bus with 5 V connection

Access: 40 HEX addresses memory space or
40 HEX addresses I/0-Space

Base address: automatically assigned by the operating system

4.1 Header Configuration

Addr.	Byte 3	Byte 2	Byte 1	Byte 0	Value (Hex)
00h	Device ID		Vendor ID		9050 10B5
18h	Base address local memory space				xxxx xxxx
1Ch	Base address local I/O space				xxxx xxxx
2C	Sub device ID		Sub vendor ID		2302 4301

A PCI interface card is identified by its four ID values; Sub device ID and Sub vendor ID only refer to IFC 430R.

4.2 Local address allocation

Base addr. +	Function	Access	Bus width
00h	System control register	Write and read	16 bits
02h	Timer register	Write and read	16 bits
04h	Status register 1	Read only	16 bits
06h	Status register 2	Read only	16 bits
08h	Interrupt register	Write and read	16 bits
0Ah	Sync-In delay timer	Write only	16 bits
0Ch	reserved		
0Eh	reserved		
10h – 1Eh	Counter channel 1	Write and read	16 bits
20h – 2Eh	Counter channel 2	Write and read	16 bits
30h – 3Eh	Counter channel 3	Write and read	16 bits

5. DESCRIPTION OF THE REGISTERS

5.1 System Control Register

Base address + 0 (write and read access)

Bit	Function
0	Activation of software sync
1	Activation of Casc Out by software
2	Activation of Casc Out by timer
3	Activation of Casc Out by sync-IN
4	Activation of IN1 as ref. pulse inhibitor for encoder channel 1
5	Activation of IN3 as ref. pulse inhibitor for encoder channel 2
6	Activation of IN5 as ref. pulse inhibitor for encoder channel 3
7	Activation of the interrupt inhibit on the PCI bus (as of hardware revision 1 only)
8	IN1 inverted
9	IN2 inverted
10	IN3 inverted
11	IN4 inverted
12	IN5 inverted
13	IN6 inverted
14	Sync-IN inverted
15	Casc-OUT inverted

5.2 Timer Register

Base address +2 (write and read access)

Write access	Read access
Timer value 0 = timer off 1 to FFFFh = timer on	Time remaining until next pulse

F = PCI frequency / 256 / (timer value +1)

5.3 Status Register 1

Base address + 4 (read access only)

Bit	Function
0	Logic level of encoder channel 1, track A (X1 pins 2 and 15)
1	Logic level of encoder channel 1, track B (X1 pins 3 and 16)
2	Logic level of encoder channel 1, track C (X1 pins 4 and 17)
3	Logic level of encoder channel 2, track A (X1 pins 6 and 19)
4	Logic level of encoder channel 2, track B (X1 pins 7 and 20)
5	Logic level of encoder channel 2, track C (X1 pins 8 and 21)
6	Logic level of encoder channel 3, track A (X1 pins 10 and 23)
7	Logic level of encoder channel 3, track B (X1 pins 11 and 24)
8	Logic level of encoder channel 3, track C (X1 pins 12 and 25)
9	Logic level of encoder interference signal (X1 pin 13)
10	Logic level I1 (X2 pin 2)
11	Logic level I2 (X2 pin 3)
12	Logic level I3 (X2 pin 4)
13	Logic level I4 (X2 pin 5)
14	Logic level I5 (X2 pin 6)
15	Logic level I6 (X2 pin 7)

5.4 Status Register 2

Base address +6 (read access only)

Bit	Function
0	1 = encoder channel 1: first reference mark traversed
1	1 = encoder channel 1: second reference mark traversed
2	1 = encoder channel 2: first reference mark traversed
3	1 = encoder channel 2: second reference mark traversed
4	1 = encoder channel 3: first reference mark traversed
5	1 = encoder channel 3: second reference mark traversed
6	Logic level Sync-IN (X2 pin 8)
7	1 = interrupt request active (as of hardware revision 1 only)
8	reserved
9	reserved
10	reserved
11	reserved
12	reserved
13	reserved
14	reserved
15	reserved

5.5 Interrupt Register

Base address +8 (write and read access)

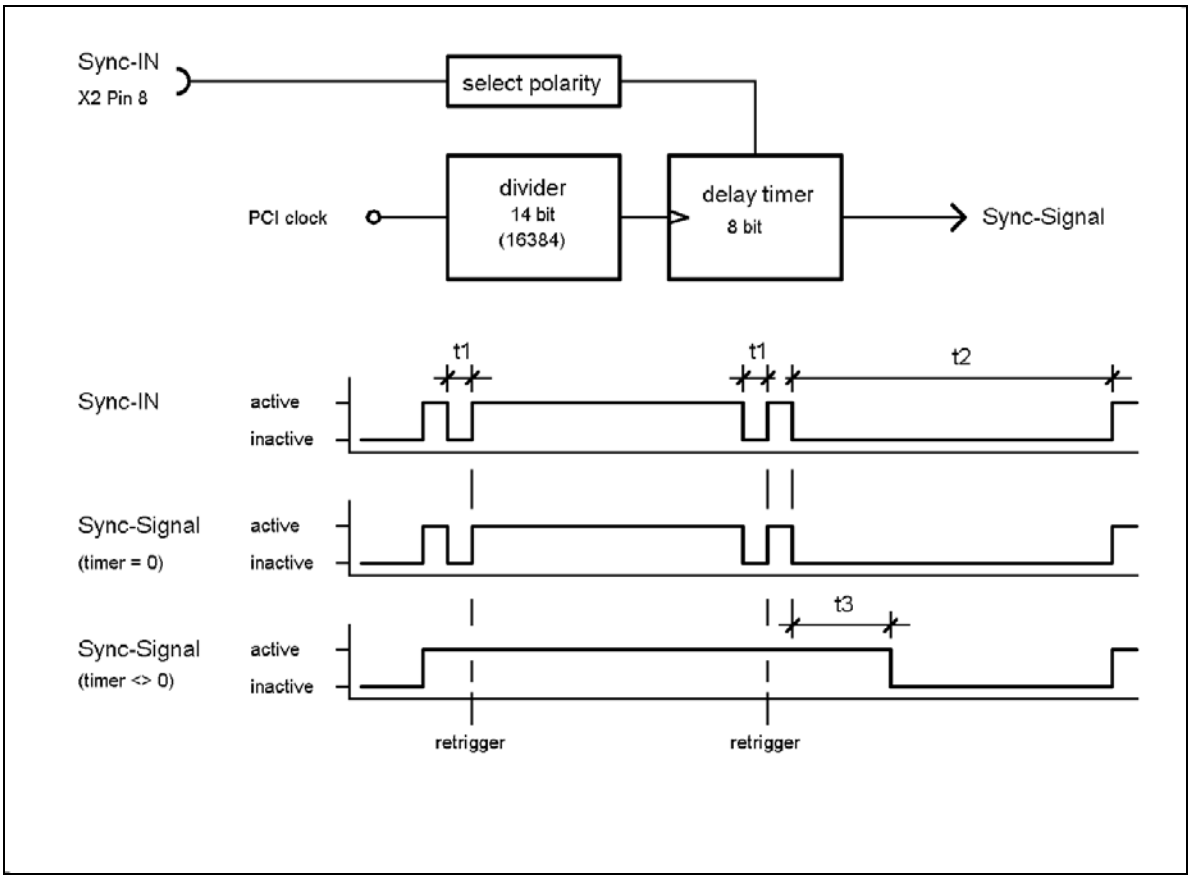
Bit	Function
0	0 = disable timer interrupt
	1 = enable timer interrupt
1	0 = disable sync interrupt
	1 = enable sync interrupt (X2 pin 8)
2	0 = disable IN1 interrupt
	1 = enable IN1 interrupt (X2 pin 2)
3	0 = disable IN2 interrupt
	1 = enable IN2 interrupt (X2 pin 3)
4	0 = disable IN3 interrupt
	1 = enable IN3 interrupt (X2 pin 4)
5	0 = disable IN4 interrupt
	1 = enable IN4 interrupt (X2 pin 5)
6	0 = disable IN5 interrupt
	1 = enable IN5 interrupt (X2 pin 6)
7	0 = disable IN6 interrupt
	1 = enable IN6 interrupt (X2 pin 7)
8	0 = disable encoder channel 1 ref1 interrupt
	1 = enable encoder channel 1 ref1 interrupt
9	0 = disable encoder channel 1 ref2 interrupt
	1 = enable encoder channel 1 ref2 interrupt
10	0 = disable encoder channel 2 ref1 interrupt
	2 = enable encoder channel 2 ref1 interrupt
11	0 = disable encoder channel 2 ref2 interrupt
	2 = enable encoder channel 2 ref2 interrupt
12	0 = disable encoder channel 3 ref1 interrupt
	3 = enable encoder channel 3 ref1 interrupt
13	0 = disable encoder channel 3 ref2 interrupt
	3 = enable encoder channel 3 ref2 interrupt
14	0 = disable error interrupt
	1 = enable error interrupt (X2 pin 25)
15	reserved

The interrupts are edge-sensitive; they are triggered when changing from inactive to active status, provided that they have been enabled by setting the corresponding bits. A read access to the interrupt register returns the active interrupt source(s).
After a read access the interrupt status is automatically cleared.

5.6 Delay Timer for external Sync-In

Base address +10 (write access only)

Write access
Timer value (8 bit)
0 = timer off, 1 to 255 = timer on



t1 < t3 < t2

t3 = timer value * T_PCI * 16384

(T_PCI = 30ns at 33.3 MHz PCI frequency)
(T_PCI * 16384 = 491.52ms at 33.3 MHz PCI frequency)

The delay timer serves to activate a drop-out delay and thus a debouncing of the Sync-In input. If the timer value is zero, the delay timer is inactive and the Sync signal directly follows the input.

If the timer value is "<> 0", the timer is triggered each time the input signal is activated. If the input signal (t2) is inactive longer than the timer (t3), the Sync signal is deactivated as soon als the timer has run off.

Notes:

The time t3 must be programmed longer than the time for signal drops (t1).

If the output "Casc Out" (X2, pin 9) is programmed such that it is triggered with the Sync signal (see chapter 5.1 System Control Rregister), it can be measured at "Casc Out".

5.7 Counter Load Register (write access)

Base addr. +	Load register
10h	Encoder channel 1 LO word
12h	Encoder channel 1 HI word
...	
20h	Encoder channel 2 LO word
22h	Encoder channel 2 HI word
..	
30h	Encoder channel 2 LO word
32h	Encoder channel 2 HI word

Note:
A write access only sets the load register. Data transfer from the load register into the counter requires a further hardware or software action (see chapter 5.9 “Counter control register” and chapter 5.10 “Counter mode register”, bits 4 to 6).

5.8 Counter Latch Register (read access)

Base addr. +	Latch register
10h	Encoder channel1 latch register 0 LO word
12h	Encoder channel1 latch register 0 HI word
14h	Encoder channel1 latch register 1 LO word
16h	Encoder channel1 latch register 1 HI word
..	
20h	Encoder channel2 latch register 0 LO word
22h	Encoder channel2 latch register 0 HI word
24h	Encoder channel2 latch register 1 LO word
26h	Encoder channel2 latch register 1 HI word
..	
30h	Encoder channel3 latch register 0 LO word
32h	Encoder channel3 latch register 0 HI word
34h	Encoder channel3 latch register 1 LO word
36h	Encoder channel3 latch register 1 HI word

Note:
Each encoder channel features two latch registers. To obtain the current count, it must be intermediately stored in one of the two latch registers (see chapter 5.9 “Counter Control Register” and chapter 5.10 “Counter mode register”, bits 8 to 10 and bits 12 to 14) by means of a hardware or software action.

5.9 Counter Control Register (write access)

Base address + 14 (encoder channel 1)
Base address +24 (encoder channel 2)
Base address +34 (encoder channel 3)

Bit	Function
0	1 = clear counter
1	1 = load counter
2	1 = store count in latch register 0
3	1 = store count in latch register 1
4	1 = enable encoder reference pulse
5 - 15	reserved

The counter control register serves to clear the counter by software or to load the contents of the load register; the current counts can be stored in one of the two latch registers.

5.10 Counter Mode Register (write / read access)

Base address +18 (encoder channel 1)
Base address +28 (encoder channel 2)
Base address +38 (encoder channel 3)

Bit	Function
0 1	**Phase discriminator** Bit 1 Bit 0 0 0 counter without phase discriminator track A = counting-direction signal track B = counter clock signal track C = counter load or latch signal 0 1 counter with phase discriminator and one-fold evaluation 1 0 counter with phase discriminator and two-fold evaluation 1 1 counter with phase discriminator and four-fold evaluation Operating mode 0 is intended for counting functions without encoders. The operating modes 1 to 3 are intended for encoder applications.
2	**Counting direction** 0 = normal counting direction 1 = inverted counting direction

3	**Inverting encoder tracks A and B** 0 = Encoder tracks A and B are not inverted Encoder ref. signal is located in the first quadrant (tracks A, B and C = 1) 1 = Encoder tracks A and B are inverted Encoder ref. signal is located in the third quadrant tracks A and B = 0, C = 1)
4 5 6	**Selecting counter load / clear signal** Bit 6 Bit 5 Bit 4 0 0 0 Hardware signals disabled 0 0 1 Clear counter with next encoder reference pulse 0 1 0 Clear counter with all encoder reference pulses 0 1 1 Clear counter with timer 1 0 0 Load counter with next encoder reference pulse 1 0 1 Load counter with all encoder reference pulses 1 1 0 Clear counter with all encoder reference pulses and additionally load counter with negative zero crossover 1 0 1 Load counter with IN 1 at X2 for encoder channel 1 IN 3 at X2 for encoder channel 2 IN 5 at X2 for encoder channel 3
7	reserved
8 9 10	**Selecting hardware signal for latch register 0** Bit 10 Bit 9 Bit 8 0 0 0 Hardware signals disabled 0 0 1 Latch counter with software sync (addr. 0 bit 0) 0 1 0 Latch counter with timer 0 1 1 Latch counter with Sync-IN at X2 1 0 0 Latch counter with IN 2 at X2 for encoder channel 1 IN 4 at X2 for encoder channel 2 IN 6 at X2 for encoder channel 3 1 0 1 Latch counter with next encoder reference pulse 1 1 0 Latch counter with second encoder reference pulse 1 1 1 Latch counter with all encoder reference pulses
11	reserved
12 13 14	**Selecting hardware signal for latch register 1** identical with bits 8 to 10
15	reserved

6. INSTRUCTIONS FOR INSTALLATION

6.1 Hardware Installation

- **For the installation please observe the safety precautions according to DIN EN 100 015 when handling ESD components (electrostatic discharge**)
- Disconnect the PC from the line by disengaging the power connector
- Open the PC
- Insert the IFC 430R interface card into a free PCI slot
- Close the PC
- Engage the power connector to the line
- Switch the PC on

6.2 Installation of the Drivers

WINDOWS XP / VISTA / 7 / 8

After the booting procedure the operating system automatically detects the new hardware component.
For this purpose the related drivers need to be installed.

- Insert the floppy disk with the label "IFC430R Driver" into the floppy-disk drive
- Follow the instructions on the screen.

By carrying out these steps an entry is made to the registry and the following files are copied to the system directory:

e. g. "C:\WINNT\System32\Drivers\IFC430R.SYS"

The enclosed disk contains the DLLs for 32 and 64-bit systems.
The corresponding DLL should be included directly in the application.

6.3 Installation of the enclosed Demo Software

The test program uses the previously installed drivers. No setup of this software is required. The demo program can be started from floppy disk or from hard disk after being copied there.

6.4 Programming Examples

On the supplied disk are programming examples for:

- Borland C-Builder 4
- Borland Delphi 5
- Microsoft Visual Basic 6
- Microsoft Visual C 6

7. DLL-FUNCTIONS

IFC_OpenDrv
Opens the device driver and returns its status.
Prototype: Data = IFC_OpenDrv();
Data: 0 = driver could not be found
1 = driver was opened

IFC_CloseDrv
Closes the device driver.
Prototype: IFC_CloseDrv();

IFC_ScanBus
Searches the PCI bus for available IFC430R, returns number of cards and their base addresses. Up to 8 cards are supported and thus 8 long words returned. For each IFC430R found the value of the related base address is not zero.
Prototype: Card = IFC_ScanBus(long*pBase Addr);
Card: Numer of cards detected (0 to 8)
Data: Pointer to 8 long words
Long word = 0 - no IFC430R was found at this position
Long word <> 0 - base address of an IFC430R

IFC_GetHwRev
Feature of hardware revision of IFC430R.

Prototyp: Revision = IFC_GetHwRev(UCHAR Card);
Revision: Hardware-Revision (0 bis 255)
Card: Nummer der Karte (0 bis 7)

IFC_Init
Initializes the IFC430R with the values from an initializing file.

Prototype: Data = IFC_Init(char *pFileName);
Data: 0 = no initializing file found
1 = initializing file was transferred
*pFileName: Pointer to the file name of the initializing file. If no file name is specified by the user, the standard initializing file is “IFC430R.INI”

IFC_ExtInit
Initializes the polarity for the external inputs and outputs at X2.
Prototype: **IFC_ExtInit(UCHAR Card, USHORT Data);**
Card: Number of the card (0 to 7)
Data: Bit 0 = 0 - input 1 low-active (X2, pin 2)
1 - " " high- "
Bit 1 = 0 - input 2 low-active (X2, pin 3)
1 - " " high- "
Bit 2 = 0 - input 3 low-active (X2, pin 4)
1 - " " high- "
Bit 3 = 0 - input 4 low-active (X2, pin 5)
1 - " " high- "
Bit 4 = 0 - input 5 low-active (X2, pin 6)
1 - " " high- "
Bit 5 = 0 - input 6 low-active (X2, pin 7)
1 - " " high- "
Bit 6 = 0 - input SyncIn low-active (X2, pin 8)
1 - " " high- "
Bit 7 = 0 - output CascOut low-active (X2, pin 9)
1 - " " high- "

IFC_ExtIn
Provides the states of the external inputs of an IFC430R.
Prototype: **Data = IFC_ExtIn(UCHAR Card);**
Card: Number of the card (0 to 7)
Data: Bit 0 = 0 - input 1 inactive (X2, pin 2)
1 - " " active
Bit 1 = 0 - input 2 inactive (X2, pin 3)
1 - " " active
Bit 2 = 0 - input 3 inactive (X2, pin 4)
1 - " " active "
Bit 3 = 0 - input 4 inactive (X2, pin 5)
1 - " " active
Bit 4 = 0 - input 5 inactive (X2, pin 6)
1 - " " active "
Bit 5 = 0 - input 6 inactive (X2, pin 7)
1 - " " active

IFC_CascInit
Initializes the output "CascOut" (X2 Pin 9).
Prototype: **IFC_CascInit(UCHAR Card USHORT Value);**
Card:Number of the card (0 to 7)
Value: 0 = CascOut inactive
1 = CascOut active
2 = CascOut switched by timer
3 = CascOut follows the input signal SyncIn

IFC_ExtSync
Provides the status of the SyncIn input (X2, pin 8).
Prototype: **SyncIn = IFC_ExtSync(UCHAR Card);**
Card: Number of the card (0 to 7)
SyncIn: 0 = SyncIn inactive
1 = " active

IFC_SetSyncDelay

Sets the preload value for the delay timer

Prototyp: **Bool = IFC_SetSyncDelay(UCHAR Card, USHORT TimerValue);**
Bool: 0 = no delay timer available with this hardware revision
1 = delay timer was set
Card: Number of the card (0 to 7)
TimerValue: 0 to 255

IFC_IntInit
Initializes the interrupts. The interrupt sources are gated with "OR"; as a consequence, several interrupt events can be processed simultaneously.
Prototype: **void IFC_IntInit(UCHAR Card, USHORT Data);**
Card: Number of the card (0 to 7)
Data: Bit 0 = 1 - interrupt enable on timer zero crossover
Bit 1 = 1 - interrupt enable on activation of SyncIn (X2, pin 8)
Bit 2 = 1 - interrupt enable on activation of IN1 (X2, pin 2)
Bit 3 = 1 - interrupt enable on activation of IN2 (X2, pin 3)
Bit 4 = 1 - interrupt enable on activation of IN3 (X2, pin 4)
Bit 5 = 1 - interrupt enable on activation of IN4 (X2, pin 5)
Bit 6 = 1 - interrupt enable on activation of IN5 (X2, pin 6)
Bit 7 = 1 - interrupt enable on activation of IN6 (X2, pin 7)
Bit 8 = 1 - interrupt enable with the next reference mark Axis 0
Bit 9 = 1 - interrupt enable with the second reference mark Axis 0
Bit 10 = 1 - interrupt enable with the next reference mark Axis 1
Bit 11 = 1 - interrupt enable with the second reference mark Axis 1
Bit 12 = 1 - interrupt enable with the next reference mark Axis 2
Bit 13 = 1 - interrupt enable with the second reference mark Axis 2
Bit 14 = 1 - interrupt enable on encoder error (X1 Pin 25)
Bit 15 = reserved

IFC_IntStatus
Returns the interrupt status. After reading out the interrupt status, all bits are deleted and set anew by the next interrupt event (edge-sensitive triggering).

Prototype: **Status = USHORT IFC_IntStatus(UCHAR Card);**
Card: Number of the card (0 to 7)
Status: Bit 0 = 1 - interrupt activation through timer zero crossover
Bit 1 = 1 - interrupt activation through SyncIn (X2, pin 8)
Bit 2 = 1 - interrupt activation through IN1 (X2, pin 2)
Bit 3 = 1 - interrupt activation through IN2 (X2, pin 3)
Bit 4 = 1 - interrupt activation through IN3 (X2, pin 4)
Bit 5 = 1 - interrupt activation through IN4 (X2, pin 5)
Bit 6 = 1 - interrupt activation through IN5 (X2, pin 6)
Bit 7 = 1 - interrupt activation through IN6 (X2, pin 7)
Bit 8 = 1 - interrupt activation through first reference mark at Axis 0
Bit 9 = 1 - interrupt activation through second reference mark at Axis 0
Bit 1 = 1 - interrupt activation through first reference mark at Axis 1
Bit 11 = 1 - interrupt activation through second reference mark at Axis 1
Bit 12 = 1 - interrupt activation through first reference mark at Axis 2
Bit 13 = 1 - interrupt activation through second reference mark at Axis 2
Bit 14 = 1 - interrupt activation through encoder error (X1 Pin 25)
Bit 15 = reserved

IFC_IntUnMask
Activation of the interrupt handler. When the interrupt handler is active the DLL runs the service routine transferred as pointer with each hardware interrupt. Different interrupt sources can be initialized (see IFC_IntInit).

Prototype:void IFC_IntUnMask(UCHAR Card, BOOL IrqShared, pIrqHandler Long);
Card: Number of the card (0 to 7)
IrqShared: True Þ Interrupt is shared
False Þ Interrupt is not shared
pIrqHandler: Pointer to interrupt service routine

Example of an interrupt service routine

```
void __stdcall IRQ_Handler(USHORT IRQ_Number, ULONG IRQ_Source)
                {
                 /* Interrupt service routine
                With every generation the interrupt handler reads the interrupt status of IFC430R and transfers it to
                the interrupt service routine as parameter (IRQ-Source). The interrupt number is also transferred as
                parameter (IRQ number 1-15).
                 */
                }
```

Notes:
currently not executable under “Windows 2000” and “Windows XP”

IFC_IntMask
Deactivation of the interrupt handler.

Prototype: **Data = IFC_IntMask(UCHAR Card);**
Card: Number of the card (0 to 7)
Data: 0 Þ Interrupt handler already inactive
1 Þ Interrupt handler switched from active to inactive status

IFC_IntIsMask
Provides the status of the interrupt handler.

Prototype: **Data = IFC_IntIsMask(UCHAR Card);**
Card: Number of the card (0 to 7)
Data: True Þ Interrupt handler inactive
False Þ Interrupt handler active

IFC_IntCnt
Read out interrupt counter.

Prototype: **Data = IFC_IntCnt(UCHAR Card);**
Card: Number of the card (0 to 7)
Data: Number (ULONG) of the interrupt routines executed since the last activation of the handler

IFC_SetTimer
Sets the preload value for the timer. If the preload value is set to "0", a running timer is stopped at the next zero crossover. If the preload value is <> 0, a non-running timer is immediately started with the specified value; a running timer receives the new value at the next zero crossover.
time = (timer value + 1) * 256 * T_PCI
timer value = preload value
T_PCI = PCI clock time (= 30ns @ 33.3MHz)
Prototype: **IFC_SetTimer(UCHAR Card, USHORT TimerValue);**
Card: Number of the card (0 to 7)
TimerValue: 0 to 65535

IFC_GetTimer
Returns the run time remaining until the next zero crossover of the timer. During the zero crossover of the timer a signal is generated that can be used for a variety of functions depending on the initializing; e.g. synchronous latching of count values, load counter or generating a pulse at the CascOut output.
Prototype: **Time = IFC_GetTimer(UCHAR Card);**
Time: 0 to 65535
Card: Number of the card (0 to 7)

IFC_SetAdr
Writes data directly to the specified address.
Prototype: **IFC_SetAdr(UCHAR Card, USHORT Offset, USHORT Data);**
Card: Number of the card (0 to 7)
Offset: 0 to 64 (address = base address of the card + offset)
Data: data (16 bits)

IFC_GetAdr
Reads data from the specified address.
Prototype: **Data = IFC_GetAdr(UCHAR Card, USHORT Offset);**
Data: 0 to 65535
Card: Number of the card (0 to 7)
Offset: 0 to 64 (address = base address of the card + offset)

IFC_EncStatus
Provides the status of the encoder inputs of an IFC430R.
Prototype: **Data = IFC_EncStatus(UCHAR Axis);**
Axis: Number of the axis (0 to 23)
Data: 0 to 7
Bit 0 = 0 - track A inactive
1 - " " active
Bit 1 = 0 - track B inactive
1 - " " active
Bit 2 = 0 - track C inactive
1 - " " active

IFC_EncErr
Provides the status of the encoder interfering signal (input X1, pin 13)
Prototype: **Data = IFC_EncErr(UCHAR Card);**
Card: Number of the card (0 to 7)
Data: 0 = encoder error signal inactive
1 = " " " active

IFC_RefInit
Initializing of external inputs as reference pulse inhibitor.
Prototype: **IFC_RefInit(UCHAR Card, USHORT Data)**
Card: Number of the card (0 to 7)
Data: 0 to 7
Bit 0 = 0 - input 1 (X2 pin 2) has no influence on the reference pulse
1 - " " acts as reference-pulse inhibitor for encoder channel 1
Bit 1 = 0 - input 3 (X2 pin 4) has no influence on the reference pulse
1 - " " acts as reference-pulse inhibitor for encoder channel 2
Bit 1 = 0 - input 5 (X2 pin 6) has no influence on the reference pulse
1 - " " acts as reference-pulse inhibitor for encoder channel 3

IFC_RefClear
Enables the reference pulses (clears the reference status).
Prototype: **IFC_RefClear(UCHAR Axis)**
Axis: Number of the axis (0 to 23)

IFC_RefStatus
Provides the status of the reference marks already traversed.
Prototype: **Data = IFC_RefStatus(UCHAR Axis)**
Axis: Number of the axis (0 to 23)
Data: 0 to 3
Bit 0 = 0 - first reference mark not traversed
1 - first reference mark traversed
Bit 1 = 0 - second reference mark not traversed
1 - second reference mark traversed

IFC_SetLoadReg
Transfers the value for the counter load register.
Prototype: **IFC_SetLoadReg (UCHAR Axis, ULONG LoadValue);**
Axis: Number of the axis (0 to 23)
LoadValue: 32-bit value

IFC_Load
The counter of an axis is loaded with the contents of the load register (by software).
A counter can also be loaded by several hardware sources (see IFC_LdClr).
Prototype: **IFC_Load (UCHAR Axis);**
Axis: Number of the axis (0 to 23)

IFC_Clear
The counter of an axis is cleared. A counter can also be cleared by several hardware sources (see IFC_LdClr).
Prototype: **IFC_Clear(UCHAR Axis);**
Axis: Number of the axis (0 to 23)

IFC_ CntLdClr
Mode to load or clear a counter by means of a hardware signal.
Prototype: **IFC_ CntLdClr (UCHAR Axis, UCHAR Value);**
Axis: Number of the axis (0 to 23)
Value: 0 to 7

0 = hardware signals disabled
1 = clear counter with next encoder reference mark
2 = clear counter with all encoder reference marks
3 = clear counter with timer
4 = load counter with next encoder reference mark
5 = load counter with all encoder reference marks
6 = clear counter with all encoder reference marks and additionally load counter with negative zero crossover
7 = load counter with
IN 1 at X2 for encoder channel 1
IN 3 at X2 for encoder channel 2
IN 5 at X2 for encoder channel 3

IFC_GetLatchReg
Provides the 32-bit count value previously stored in a latch register.
Prototype: **Data = IFC_GetLoadReg (UCHAR Axis, UCHAR Reg);**
Data: 32-bit counter value
Axis: Number of the axis (0 to 23)
Reg: Number of the latch register (0 or 1)

IFC_Latch
Stores the 32-bit count value of an axis in one of the two latch registers (by software). The count value can also be stored in the latch registers by means of several hardware sources (see IFC_CntLatch).
Prototype: **IFC_Latch (UCHAR Axis, UCHAR Reg);**
Axis: Number of the axis (0 to 23)
Reg: Number of the latch register (0 or 1)

IFC_LatchImp
Generates a hardware pulse that is simultaneously fed to all latch registers of a card; thus, several count values of a card can be latched synchronously by software. The pulse can also be switched to the CascOut output (X2, pin 9) at the same time. If the CascOut of this card is connected to the SyncIn of the next cards, several count values of several cards can be latched synchronously by software.
Prototype: **IFC_LatchImp (UCHAR Card, UCHAR Casc);**
Card: Number of the card (0 to 7)
Casc: 0 = latch pulse is not fed to CascOut (X2, pin 9)
1 = latch pulse is simultaneously fed to CascOut (X2, pin 9)

IFC_ CntLatch0
Selecting hardware signal for latch register 0
Prototype: **IFC_ CntLatch0 (UCHAR Axis, UCHAR Value);**
Axis: Number of the axis (0 to 23)
Value: 0 to 7
0 = hardware signals disabled
1 = latch counter with software (function IFC_LatchImp)
2 = latch counter with timer
3 = latch counter with Sync-IN at X2
4 = latch counter with
IN 2 at X2 for encoder channel 1
IN 4 at X2 for encoder channel 2
IN 6 at X2 for encoder channel 3
5 = latch counter with next encoder reference mark
6 = latch counter with second encoder reference mark
7 = latch counter with all encoder reference marks

IFC_ CntLatch1
Selecting hardware signal for latch register 1
Prototype: **IFC_ CntLatch1 (UCHAR Axis, UCHAR Value);**
Axis: Number of the axis (0 to 23)
Value: 0 to 7

0 = hardware signals disabled
1 = latch counter with software (function IFC_LatchImp)
2 = latch counter with timer
3 = latch counter with Sync-IN at X2
4 = latch counter with
IN 2 at X2 for encoder channel 1
IN 4 at X2 for encoder channel 2
IN 6 at X2 for encoder channel 3
5 = latch counter with next encoder reference mark
6 = latch counter with second encoder reference mark
7 = latch counter with all encoder reference marks

IFC_ CntMode
Setting of phase discriminator, counting direction and encoder quadrants.
Prototype: **IFC_ CntMode (UCHAR Axis, UCHAR Value);**
Axis: Number of the axis (0 to 23)
Value: 0 to 15

Bit 1	Bit 0	Phase discriminator
0	0	Counter without phase discriminator track A = counting-direction signal track B = counter clock signal track C = counter load or latch signal
0	1	Counter with phase discriminator and one-fold evaluation
1	0	Counter with phase discriminator and two-fold evaluation
1	1	Counter with phase discriminator and four-fold evaluation

Operating mode 0 is intended for counting functions without encoders.
The operating modes 1 to 3 are intended for encoder applications.

Bit 2	Counting direction
0	Normal counting direction
1	Inverted counting direction

Bit 3	Inverting encoder tracks A and B
0	Encoder tracks A and B are not inverted Encoder ref. signal is located in the first quadrant (tracks A, B and C = 1)
1	Encoder tracks A and B are inverted Encoder ref. signal is located in the third quadrant (tracks A and B = 0, C = 1)

DISTRIBUTION CONTACTS

AUSTRIA *Corporate Head Quarters*	**RSF Elektronik Ges.m.b.H.**	A-5121 Tarsdorf 93	☎ +43 62 78 81 92-0 FAX +43 62 78 81 92-79	e-mail: info@rsf.at internet: www.rsf.at
FRANCE	**HEIDENHAIN FRANCE sarl**	2 Avenue de la Christallerie 92310 Sèvres	☎ +33 1 41 14 30 00 FAX +33 1 41 14 30 30	e-mail: info@heidenhain.fr internet: www.heidenhain.fr
GREAT BRITAIN	**HEIDENHAIN (GB) Ltd.**	200 London Road Burgess Hill West Sussex RH15 9RD	☎ +44 1444 247711 FAX +44 1444 870024	e-mail: sales@heidenhain.co.uk internet: www.heidenhain.co.uk
ITALY	**HEIDENHAIN ITALIANA S.r.l.**	Via Asiago, 14 20128 Milano (MI)	☎ +39 02 27075-1 FAX +39 02 27075-210	e-mail: info@heidenhain.it internet: www.heidenhain.it
NETHERLANDS	**HEIDENHAIN NEDERLAND B.V.**	Copernicuslaan 34 6710 BB EDE	☎ + 31 318 58 18 00 FAX + 31 318 58 18 70	e-mail: verkoop@heidenhain.nl internet: www.heidenhain.nl
SWEDEN	**HEIDENHAIN Scandinavia AB**	Storsätragränd 5 SE-12739 Skärholmen	☎ +46 8 531 933 50 FAX +46 8 531 933 77	e-mail: sales@heidenhain.se internet: www.heidenhain.se
SWITZERLAND	**RSF Elektronik (Schweiz) AG**	Vieristrasse 14 CH-8603 Schwerzenbach	☎ +41 44 955 10 50 FAX +41 44 955 10 51	e-mail: info@rsf.ch internet: www.rsf.ch
CHINA	**RSF Elektronik**	Tian Wei San Jie, Area A, Beijing Tianzhu Airport Industrial Zone Shunyi District, 101312 Beijing P.R. China	☎ +86 10 80 42 02 88 FAX +86 10 80 42 02 90	e-mail: cao.shizhi@rsf.cn internet: www.rsf.cn
JAPAN	**HEIDENHAIN K.K.**	Hulic Kojimachi Bldg., 9F 3-2 Kojimachi, Chiyoda-ku Tokyo, 102-0083	☎ +81 3 3234 7781 FAX +81 3 3262 2539	e-mail: sales@heidenhain.co.jp internet: www.heidenhain.co.jp
KOREA	**HEIDENHAIN LTD.**	202 Namsung Plaza, 9th Ace Techno Tower, 130, Digital-Ro, Geumcheon-Gu, Seoul, Korea 153-782	☎ +82 2 20 28 74 30	e-mail: info@heidenhain.co.kr internet: www.rsf.co.kr
USA	**HEIDENHAIN CORPORATION**	333 East State Parkway Schaumburg, IL 60173-5337	☎ +1 847 490 11 91	e-mail: info@heidenhain.com internet: www.rsf.net

Date 02/2013 ■ Art.Nr.824550-01 ■ Dok.Nr. D824550-00-A-01 ■ Technical adjustments in reserve!